ABSTRACT

ANALYSIS OF MHPDM ALGORITHM FOR DATA HIDING IN JPEG IMAGES

by
Pooja Gore

In the recent years, there has been a great deal of interest in developing a secure algorithm for hiding information in images, or steganography. There has also been a lot of research in steganalysis of images, which deals with the detection of hidden information in supposedly natural images. The first section of this thesis reviews the steganography algorithms and steganalysis techniques developed in the last few years. It discusses the breadth of steganographic algorithms and steganalytic techniques, starting with the earliest, based on LSB flipping of the DCT coefficients, to more recent and sophisticated algorithms for data hiding and equally clever steganalytic techniques.

The next section focuses on the steganographic algorithm, MHPDM which was first developed by Eggers and then modified by Tzschoppe, Bauml, Huber and Kaup. The MHPDM algorithm preserves the histogram of the stego image and is thus perfectly secure in terms of Cachin's security definition. The MHPDM algorithm is explained in detail and implemented in MATLAB. It is then tested on numerous images and steganalysed using Dr. Fridrich's recent feature-based steganalytic technique. The thesis concludes with observations about the detectibility of MHPDM using feature-based steganalysis for different payloads (embedded message lengths).

ANALYSIS OF MHPDM ALGORITHM FOR DATA HIDING IN JPEG IMAGES

by
Pooja Gore

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

August 2005

APPROVAL PAGE

ANALYSIS OF MHPDM ALGORITHM FOR DATA HIDING IN JPEG IMAGES

Pooja Gore

Dr. Ali Akansu, Thesis Advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Richard A Haddad, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Taha Sencar, Committee Member — Date
Postdoctoral Researcher,
Department of Computer and Information Science,
Polytechnic University, NY

BIOGRAPHICAL SKETCH

Author: Pooja Gore

Degree: Master of Science

Date: August 2005

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2005

- Bachelor of Engineering in Instrumentation and Control,
  Cummins College of Engineering, Pune, India, 2000

Major: Electrical Engineering

To my parents, Prakash and Kunda, for their love and belief

and to my husband, Pradeep, for his complete support and encouragement

ACKNOWLEDGMENT

I would like to express deep gratitude to my thesis advisor, Dr Ali Akansu, for his ideas, constant encouragement and guidance in all phases of my graduate studies at NJIT.

Special thanks are given to Dr. Taha Sencar and Professor Richard Haddad for actively participating in my committee. Their advice and patience is appreciated.

I would also like to thank my parents and my husband for all their support.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Steganography is the art of secret or covert communication in which the very presence of a message is hidden along with its contents. Embedding the message in some multimedia data (cover data) such as images, sound or video hides the presence of a message. Steganography aims at embedding data in a way such that the altered data is perceptually the same as the cover data.

Steganalysis on the other hand, seeks to analyze the cover data and detect the presence of the embedded message. If an algorithm exists, which can guess whether or not the given cover data contains a hidden message with a success rate better than random guessing, the steganographic system is considered broken.

According to B.Pfitzmann's standard information hiding terminology [18] a stegosystem consists of two parties, Alice and Bob, who are the users of the stegosystem. Alice wishes to send an innocent-looking message with a hidden meaning over a public channel to Bob, such that the third party, the adversary Eve does not detect the hidden information.

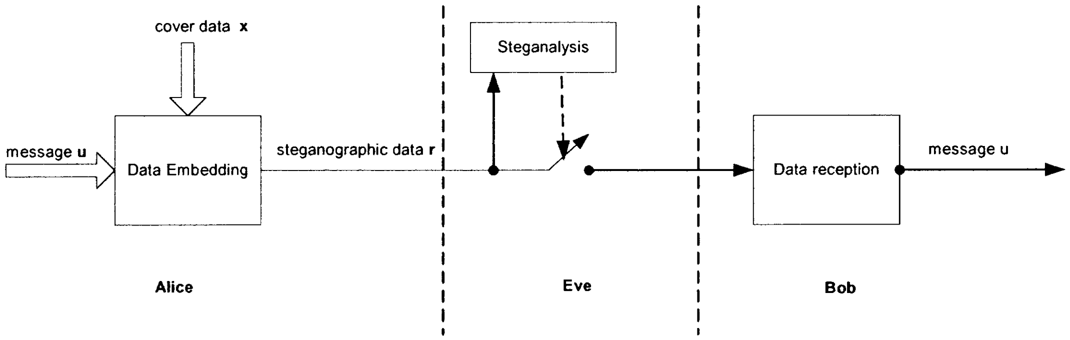


Figure 1.1 Dataflow in a stegosystem.

This thesis begins with a review of the most important steganography algorithms and steganalytic techniques developed in the last few years. Chapter 2 describes in greater detail, Eggers's HPDM (Histogram Preserving Data Mapping) algorithm [1], which is a recent steganography algorithm for JPEG images and its modified version, MHPDM. The design rules and implementation details of MHPDM are explained, followed by the experimental results for the MHPDM implementation in Chapter 3 and 4. Chapter 5 discusses the performance of the MHPDM algorithm and the motivation for steganalyzing MHPDM using Dr. J. Fridrich's recent feature-based steganalytic method [9]. It further reviews the feature-based steganalytic method, describes the experimental setup for steganalysis of MHPDM and then analyses the obtained results. The thesis concludes with observations about the effect of embedding rate or payload size on the detectibility of MHPDM embedded images by feature-based steganalysis.

CHAPTER 2

A REVIEW OF STEGNOGRAPHY ALGORITHMS AND STEGANALYSIS TECHNIQUES

2.1 Steganalysis by "Chi-Square Attack"

The Chi-Square Attack by Westfield [10] was one of the first general steganalytic methods. The original version of this method could detect sequentially embedded messages and it was later generalized to randomly scattered messages. This approach is based solely on the first order statistics and can be applied only to the early steganography algorithms like LSB (Least Significant Bit) flipping. An example of an algorithm embedding in the LSB of DCT coefficients is Jsteg.

2.2 Steganography with Histogram Preserving Data Mapping (HPDM)

The HPDM works by altering a subset of the DCT coefficients of a JPEG compressed image. This algorithm preserves in a statistical sense, the histogram of the cover data. This algorithm and its modified version are described in detail in Chapter 3.

In [2], Cachin defined the security of a steganograpic system in an information-theoretic way. He postulated for a perfectly secure system that the relative entropy of the cover data and the stego data is zero. His concept of security is based on the probability distributions of cover and stego data and assumed independent and identically distributed cover data elements. HPDM is perfectly secure in terms of Cachin's security definition.

2.3 Steganalysis using Higher-order Statistical Models

Steganalysis approaches previous to this technique typically examined first-order statistical distributions of intensity or transform coefficients. Therefore simple counter-measures to match first-order statistics could foil detection. This steganalysis method pioneered by Memon and Farid [12, 13] is based on building higher-order statistical models for natural images and looking for deviations from these models. Support vector machines (linear and non-linear) are employed to detect the alteration of higher-order statistics within a wavelet-like decomposition.

The images are decomposed using separable quadrature mirror filters (QMFs). Given this image decomposition, the statistical model is composed of the mean, variance, skewness and kurtosis.

2.4 Steganography using Modified HPDM (MHPDM)

The HPDM algorithm was perfectly secure in terms of Cachin's security definition. In [7], the authors modify HPDM such that perfect security with Farid's method described in Section 1.2 is achieved. HPDM embeds into DCT coefficients regardless of whether higher frequency components are present in the block. The reason is the structure of the embedding scheme, which treats the DCT channels as parallel and independent subchannels. The embedding distortion per subchannel due to switched data mapping is distributed over the whole subchannel. Therefore blocks containing low frequency or DC components are not treated separately and data is embedded there.

The modified HPDM or the MHPDM states that the DCT coefficients with values equal to –1, 0 and 1 should be ignored in the mapping process and thus DCT coefficients with these values are not modified.

2.5 Steganalysis Based on the Concept of a Distinguishing Statistic

In this approach by J. Fridrich [11], the steganalyst first carefully inspects the embedding algorithm and then identifies a quantity (the distinguishing characteristic) that changes predictably with the length of the embedded message, but can be calibrated for cover images. For JPEG images this calibration is done by decompressing the stego image, cropping by a few pixels in each direction, and recompressing using the same quantization table.

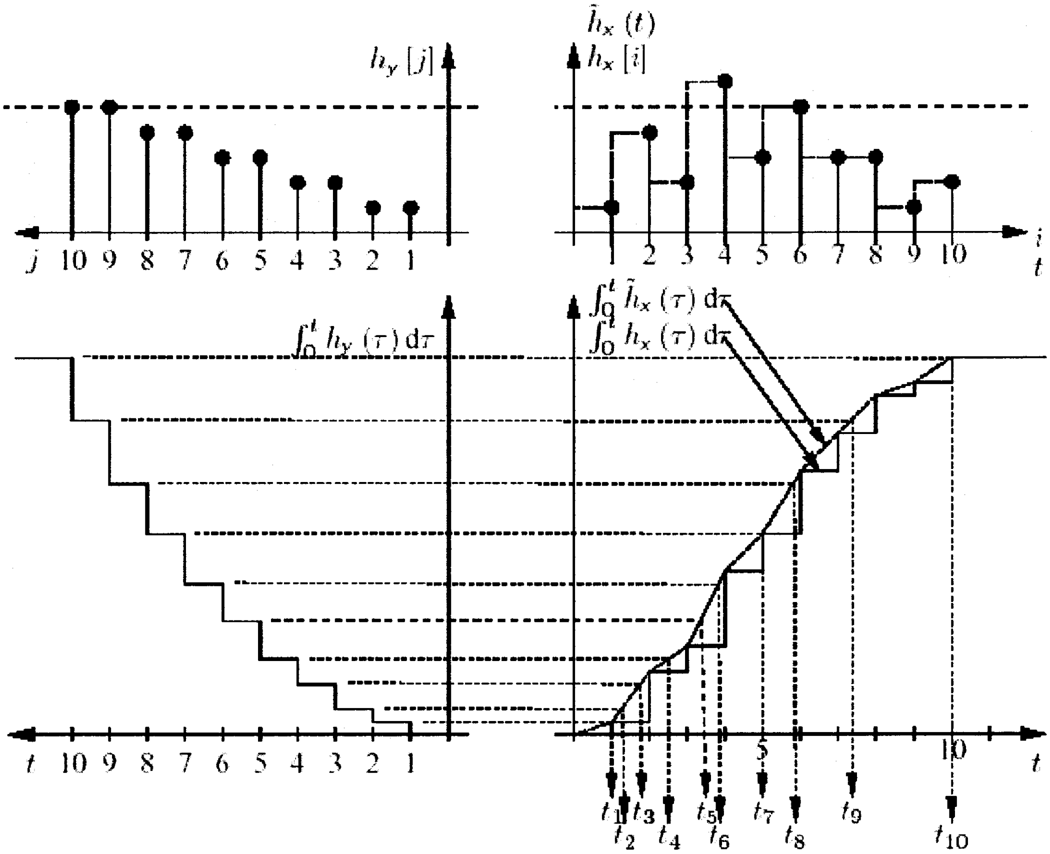
J. Fridrich subsequently combines this concept of calibration with feature based classification to device a blind detector specific to JPEG images [9].The features are calculated directly in the DCT domain and thus, detection is made more sensitive to wider types of embedding algorithms because the calibration process increases the feature's sensitivity to the embedding modifications while suppressing variations.

CHAPTER 3

DETAILED DESCRIPTION OF MHPDM ALGORITHM

3.1 Data Mapping to Achieve a Predefined Histogram

Eggers [1] proposes an efficient implementation of the random mapping $x_n \rightarrow y_n$ which involves randomizing the input data x_n and quantizing this randomized input data to the output data y_n. The mapping is characterized completely by the scalar quantizer Q_t, which is itself characterized by the set $T = \{t_1, t_2,..., N_{x-1}\}$ of decision thresholds.



$h_y[j]$
j
10 9 8 7 6 5 4 3 2 1
$\bar{h}_x(t)$
$h_x[i]$
1 2 3 4 5 6 7 8 9 10
i
t
$\int_0^t h_y(\tau)\,d\tau$
$\int_0^t \bar{h}_x(\tau)\,d\tau$
$\int_0^t h_x(\tau)\,d\tau$
t 10 9 8 7 6 5 4 3 2 1
5
10
t
t_1 t_3 t_5 t_7 t_9
t_2 t_4 t_6 t_8 t_{10}

Figure 3.1 Derivation of thresholds $\{t_1, t_2,..., N_x\}$ for the data mapping x → y.
Source: J.J.Eggers, R.Bauml and B.Girod, "A Communications Approach to Image Steganography," in Proc. Of SPIE vol. 4675, Security and Watermarking of Multimedia Contents IV, (San Jose, CA, USA), January 2002.

3.2 Information Embedding based on Switched Data Mapping

The data mapping described in Section 4.1 is used for information embedding where mappings $x_n \rightarrow r_n$ are defined and the data to be embedded is used to switch between the possible mapping rules. The information embedding is based on a principle called quantization index modulation (QIM), as proposed by Chen and Wornell [3]. Thus for information embedding, the cover data x has to be mapped onto members from disjoint sets for different possible secret messages u.

Two disjoint sets X_0 and X_1 are defined where $X_0 \cup X_1 = X$ and $X_0 \cap X_1 = 0$. These sets X_0 and X_1 are interpreted as the representatives of two different quantizers. The message u is encoded into a binary stream, b and is embedded into x by the mapping $x_n \rightarrow r_n$ using the mappings Map(X, X_0) and Map(X, X_1) for $b_n = 0$ and $b_n = 1$, respectively.

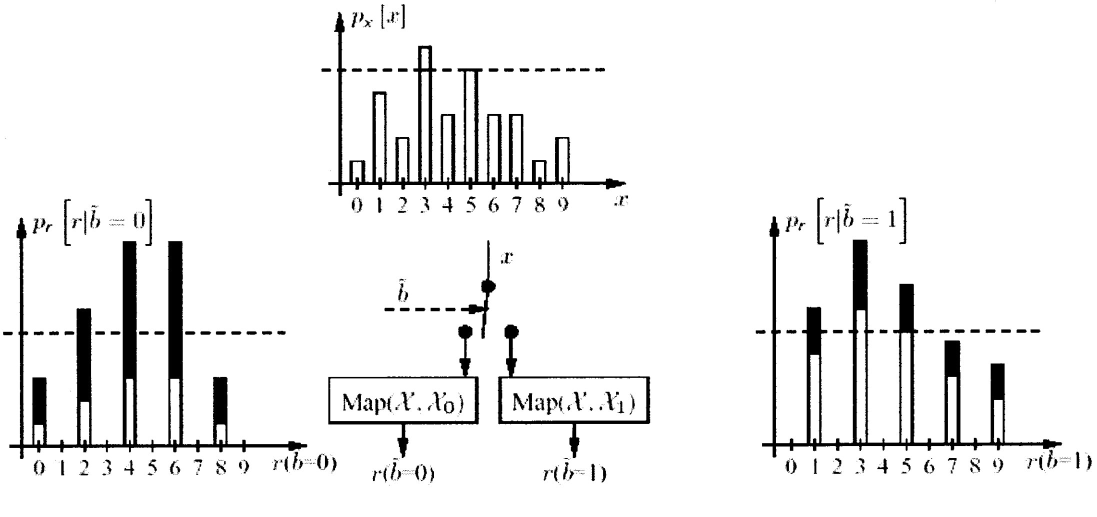


Figure 3.2 Switched data mapping for message b.
Source: J.J.Eggers, R.Bauml and B.Girod, “A Communications Approach to Image Steganography,” in Proc. Of SPIE vol. 4675, Security and Watermarking of Multimedia Contents IV, (San Jose, CA, USA), January 2002.

The mapping rules are designed such that the conditional PMFs $p_r[r \mid b = 0]$ and $p_r[r \mid b = 1]$ are scaled proportional to the cover PDF $p_x[x]$ for all members of set X_0 and X_1, respectively, and zero elsewhere.

The PMF of the cover data is not modified by the information embedding scheme if the probability Prob $(b = 1)$ of number of "1" bits in the binary message b is equal to the probability that the elements of cover data x belong to the set X_1.

According to the modification of MHPDM algorithm in [7], the DCT coefficients with values −1, 0 and 1 are ignored in the mapping process. Thus, the implemented algorithm is MHPDM.

CHAPTER 4

THE MHPDM IMPLEMENTATION

4.1 Experimental Setup

A two-dimensional Discrete Cosine Transform (DCT) is performed of non-overlapping 8 x 8 blocks of the image pixels. Each 8 x 8 block of image pixels is transformed into 64 DCT coefficients. There are i such 8 x 8 blocks with each block containing j = 64 number of DCT coefficients. The coefficients with identical frequency index j from all 8 x 8 blocks compose a subchannel. Thus there are 64 subchannels, all having the same length L_x which is identical to the number of 8 x 8 blocks in the given image. The subchannels are labeled according to the zig-zag scan.

Each subchannel is modeled by an IID random process. Each subchannel is quantized according to JPEG compression with quality factor 75. The subchannel numbers 1 to 21 in zig-zag scan were used for information embedding.

The MHPDM (modified histogram preserving data mapping) algorithm for data hiding in JPEG images was studied in detail and implemented in matlab. The publicly available JPEG source code written by Arno Swart (swart@math.uu.nl) is used in the implementation of MHPDM. The MHPDM algorithm was rigorously tested by running 100 trials using pseudo-random data as input. The relative entropy between the original histogram and the embedded histogram was calculated for 100 trials of pseudo random data and plotted in Figure 4.1. The relative entropy was found to be very small (to the order of 10^-3).

4.2 Results

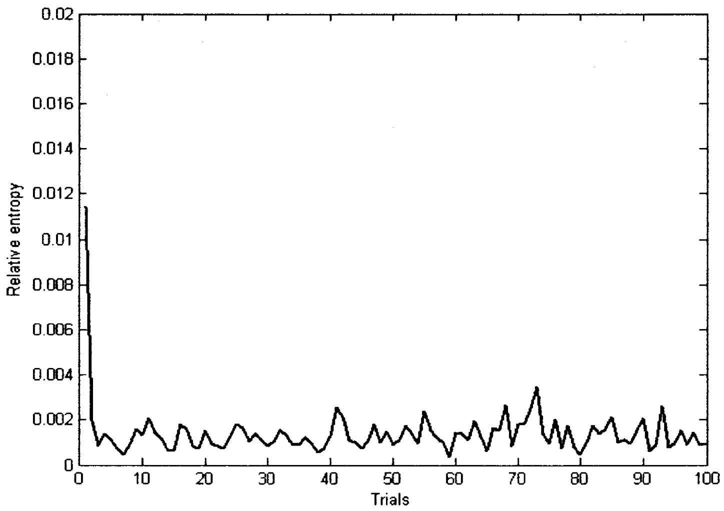


Figure 4.1 Relative entropy between original and embedded histogram of random pseudo-data for 100 trials using MHPDM.

Next, binary messages with equal number of zeros and ones were embedded into subchannels 1 to 21 of several different 512 x 512 grayscale images. The USC-SIPI image database at http://sipi.usc.edu/database was used for the images. The relative entropy between the original and embedded subchannel was calculated for subchannels 1 to 21 of every image and plotted. Sample plots for 4 images are shown in Figure 4.2, 4.3 and 4.4.

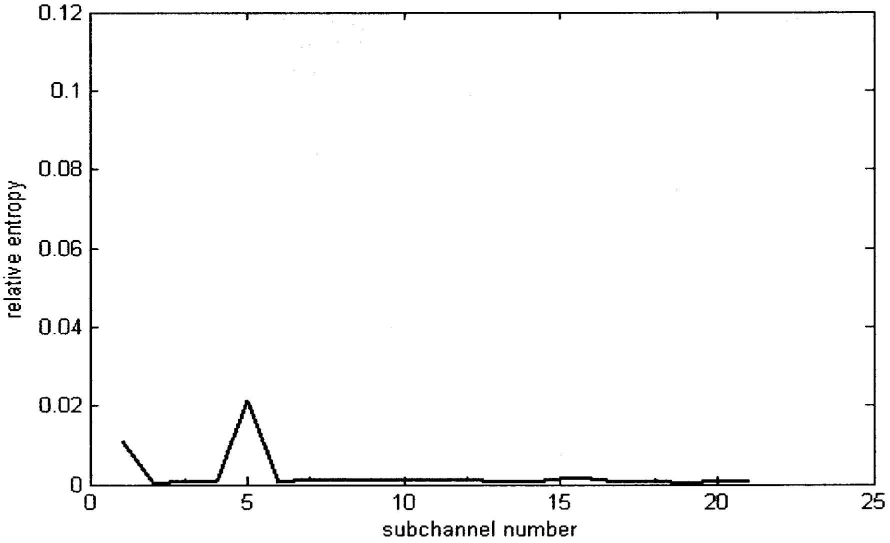


Figure 4.2 Relative entropy between original and embedded subchannels for *lenna.tiff* using MHPDM.

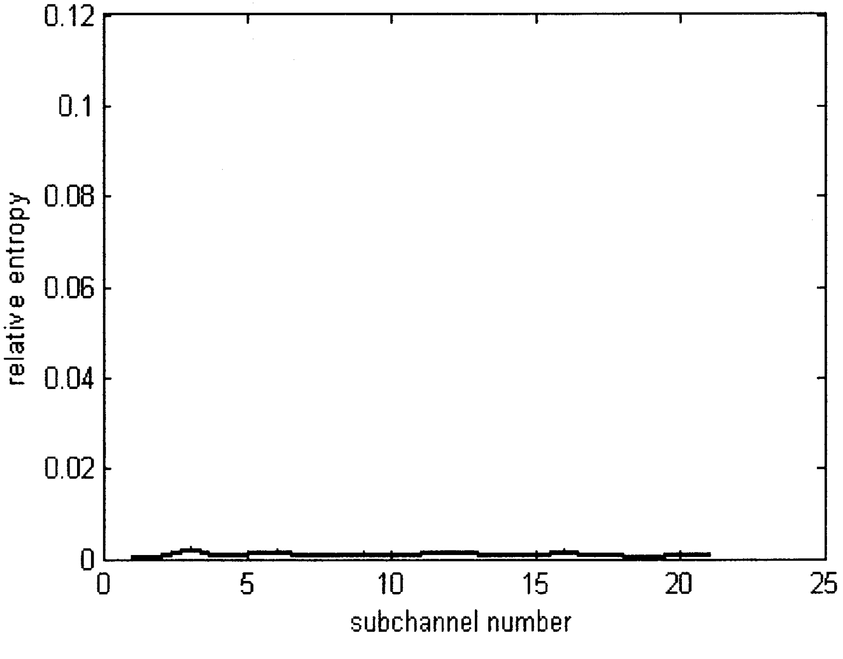


Figure 4.3 Relative entropy between original and embedded subchannels for *baboon.tiff* using MHPDM.

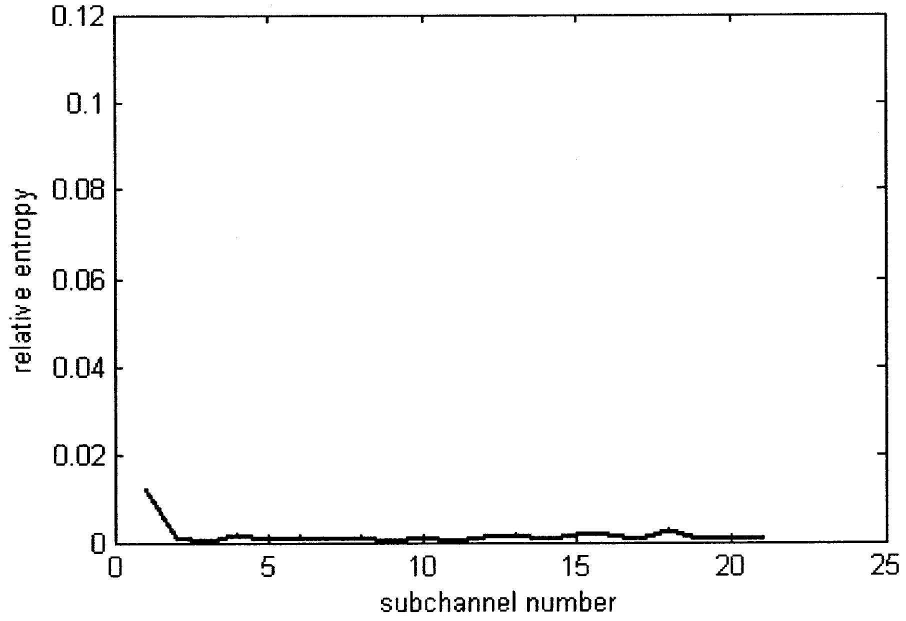


Figure 4.4 Relative entropy between original and embedded subchannels for *tiffany.tiff* using MHPDM.

The histogram of a single subchannel, subchannel 15 of *lenna.tiff* was plotted for values before and after message-embedding. These histograms shown in Figure 4.4 prove that HPDM algorithm maintains the first order statistics, i.e. the histogram of the stego image after data hiding.

Figure 4.5 shows an example of an image, peppers.jpg with different embedding rates. Note that visually, there is hardly any difference between the cover image with 0 % embedding and the stego images when viewed by a casual viewer.

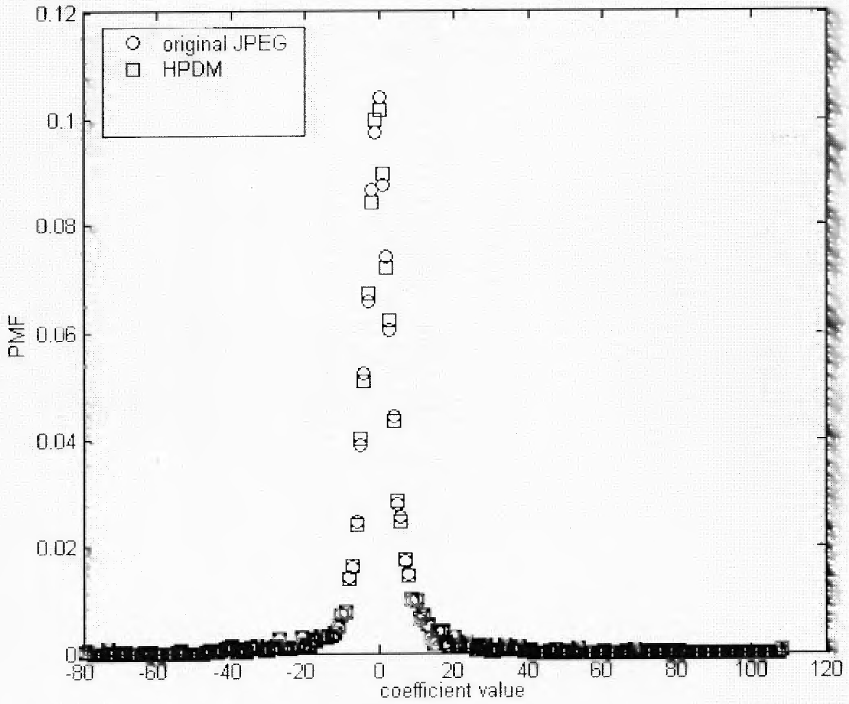


Figure 4.5 Histogram of 15th DCT subchannel of *lenna.tiff*.

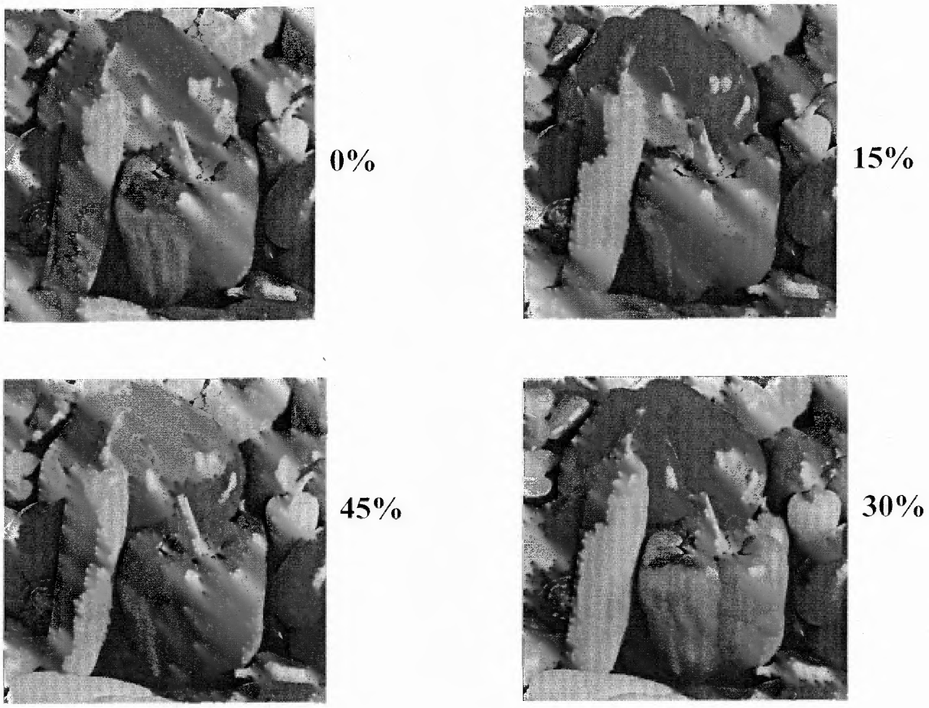


Figure 4.6 Comparison between stego images with different embedding rates

CHAPTER 5

FEATURE-BASED STEGANALYSIS OF MHPDM

5.1 Motivation

The MHPDM algorithm for image steganography preserves the histogram of the stego image after data-hiding. Thus, it is considered secure with respect to Cachin's definition of security within the given stochastic data model.

The stochastic data model assumed for development of the MHPDM algorithm models each DCT subchannel in an image by an IID random process. This model is not very accurate because in natural images, there are bound to be dependencies between different subchannels and different elements within one subchannel. Entropy encoding within JPEG compression takes advantage of dependencies between different DCT coefficients, whereas MHPDM breaks these dependencies.

J. Fridrich's feature-based steganalysis for JPEG images [9], introduced in Section 1.5, compares JPEG steganographic algorithms and further evaluates their embedding mechanisms and detectibility. This detection method is a linear classifier trained on feature vectors corresponding to cover and stego images. The features are calculated as an L_1 norm of the difference between a specific macroscopic functional calculated from a stego image and the same functional obtained from a decompressed, cropped and recompressed stego image. Dr. Fridrich tests the feature-based detection scheme on three steganographic algorithms, Outguess [14], F5 [16] and Model Based Steganography [15, 17] and concludes that all three algorithms are detectible.

In this thesis, the MHPDM algorithm for data hiding in jpeg images is steganalyzed using the feature-based steganalytic method.

5.2 Review of Feature-Based Steganalytic Method

This steganalytic method proposed by Jessica Fridrich [9] is a linear classifier trained on feature vectors corresponding to cover and stego images. There are two types of feature vectors – first order and second order features. All features are constructed by first applying a vector functional to the stego jpeg image, then decompressing it to the spatial domain, cropping by 4 pixels in either direction, recompressing it with the same quantization table, and then applying the same vector functional on this decompressed, cropped and recompressed image. The final feature f is then obtained as the L_1 norm of the difference between the two functionals.

The cropping and recompression produces a 'calibrated' image which is perceptually similar to the original cover image. The features in the feature vector are based on 23 functionals - global histogram, individual histograms for 5 DCT modes, (2,1), (3,1), (1,2), (2,2) and (1,3), dual histograms for 11 DCT values, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, variation, L_1 and L_2 blockiness and co-occurrence, N_{00}, N_{01} and N_{11}.

5.3 Experimental Setup

The USC-SIPI image database at http://sipi.usc.edu/database was used as a source of uncompressed .tiff images. All images were converted to grayscale and a quality factor of 75% was used for the JPEG compression. Messages of three different message lengths were embedded in 460 images. The message lengths were proportional to the number of DCT coefficients excluding coefficients with values of 0, 1 and –1, as MHPDM prohibits change in these coefficients. The message lengths used were:

x1 = 0.005* number of DCT coefficients excluding 0, 1 and –1 x2 = 0.05 * number of DCT coefficients excluding 0, 1 and –1

x3 = 0.15 * number of DCT coefficients excluding 0, 1 and –1

x4= 0.30 * number of DCT coefficients excluding 0, 1 and –1

x5= 0.45 * number of DCT coefficients excluding 0, 1 and –1

x6= 0.90 * number of DCT coefficients excluding 0, 1 and –1

Feature vectors were calculated for six sets of 460 stego images with the different embedding rates and 460 cover images. Instead of the Fisher Linear Discriminant classifier, used for classification in [9], we use the more sophisticated classifier, Support Vector Machines by Chih-Chung Chang and Chih-Jen Lin at http://www.csie.ntu.edu.tw/~cjlin/libsvm. The SVM classifier is trained with 660 feature vectors from random stego and cover images for every embedding rate and then the trained classifier is tested with the remaining feature vectors. The output of the SVM classifier is the percentage accuracy in detecting the stego and cover images. The percentage accuracy of detection is averaged over 10 trials with random training data set for each embedding rate. This accuracy is used to calculate the probability of success, P{success}, miss probability, P{miss}, probability of false positives, P{false positives} and probability of false negatives, P{false negatives} for all six embedding rates. Observations are made based on these values and conclusions drawn.

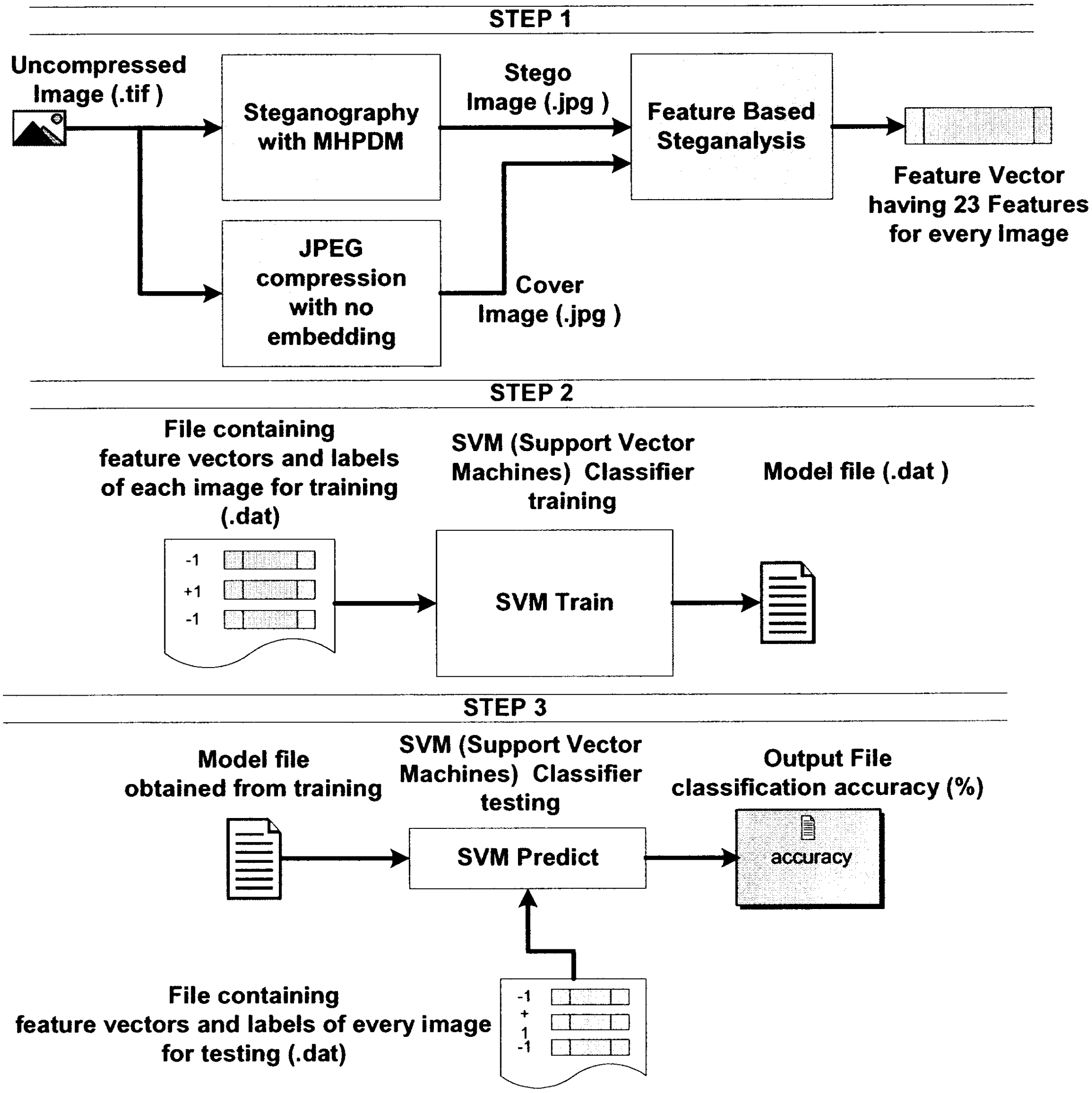

STEP 1
Uncompressed Image (.tif)
Steganography with MHPDM
Stego Image (.jpg)
Feature Based Steganalysis
Feature Vector having 23 Features for every Image
JPEG compression with no embedding
Cover Image (.jpg)
STEP 2
File containing feature vectors and labels of each image for training (.dat)
-1
+1
-1
SVM (Support Vector Machines) Classifier training
SVM Train
Model file (.dat)
STEP 3
Model file obtained from training
SVM (Support Vector Machines) Classifier testing
SVM Predict
Output File classification accuracy (%)
accuracy
File containing feature vectors and labels of every image for testing (.dat)
-1
+
1
-1

Figure 5.1 Block diagram of the experimental setup for steganalysis of MHPDM

5.4 Results of Steganalysis

The percentage accuracy values for detection of stego images for all six data embedding rates were obtained using the set up described in the previous Section. The average accuracy was calculated from 10 trials for each set.

Table 5.1 Classification Accuracy for Different Embedding Rates as an Average of 10 Random Trials.

Embedding Rate/ Trials	0.5 %	5 %	15 %	30 %	45 %	90 %
1	53.8776	65.7143	73.7288	73.5099	78.0731	84.898
2	48.1633	66.5306	72.8814	74.5033	76.0797	78.7755
3	50.6122	59.5918	73.7288	71.8543	76.079	80.4082
4	52.6531	63.2653	68.6441	73.1788	74.4186	77.9592
5	57.1429	64.0816	67.7966	71.8543	73.4219	78.3673
6	50.7143	61.6327	69.4915	76.4901	72.4252	80.4082
7	49.7959	60.8163	70.339	75.1656	79.0698	78.3673
8	52.6531	66.5306	70.7627	71.8543	76.412	82.0408
9	50.2041	63.2653	67.3729	76.589	78.0303	80.0
10	50.9388	62.8571	72.4576	76.4901	76.7442	84.0816
Average Accuracy(%)	**50.7555**	**63.428**	**70.720**	**74.105**	**76.075**	**80.530**

A ROC plot of the average percentage accuracy of detection verses the embedding rate was plotted as shown in Figure 5.1.

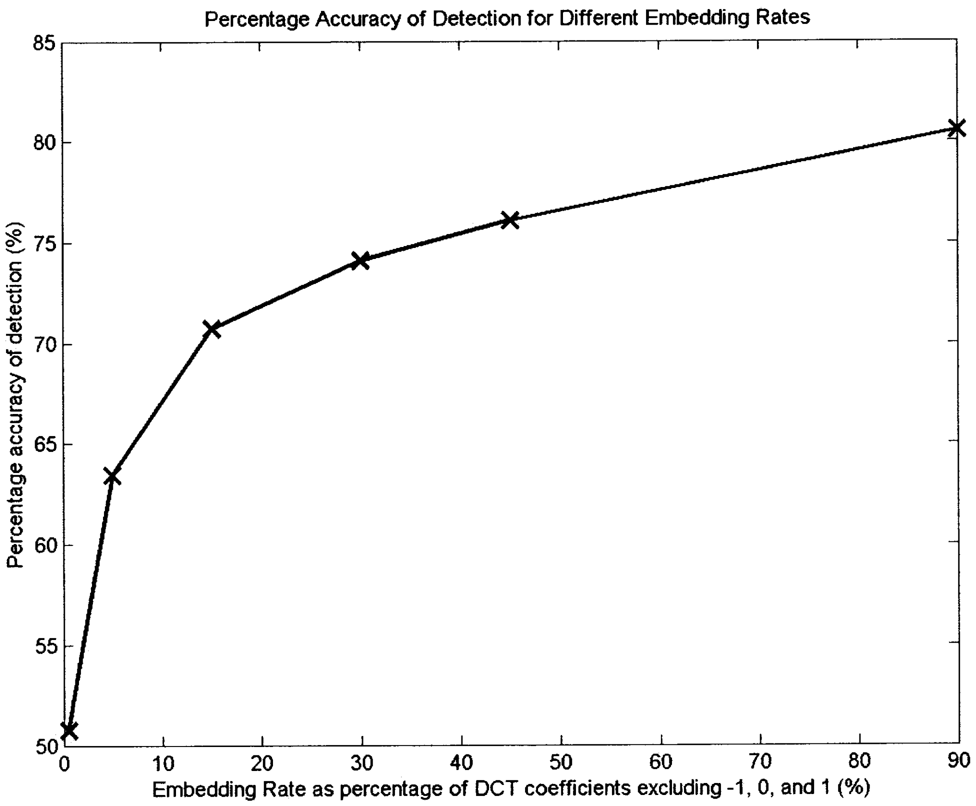


Figure 5.2 Plot of average percentage accuracy of detection for different embedding rates

CHAPTER 6

CONCLUSION

The MHPDM algorithm for data hiding in JPEG images was implemented and tested on many images. The MHPDM algorithm for image steganography preserves the histogram of the stego image after data-hiding. Thus, it can be considered perfectly secure with respect to Cachin's definition of security.

This MHPDM algorithm was then steganalyzed using the feature-based steganalytic method and the SVM (Support Vector Machines) classifier. The effectiveness of the feature-based steganalytic method for MHPDM embedded stego images with different message lengths was studied. Experiments were performed on stego images with message lengths ranging from 0.5 % to 90% of number of DCT coefficients excluding those with values of –1, 0 or 1. The accuracy results from the SVM classifier and their ROC plot in Figure 5.2 show that the classification accuracy of detection of stego and cover test images increases with increase in embedding rate. The classification accuracy of detection for 0.5 % embedding rate is an average of 50.755 % which means that MHPDM embedded stego images with a low embedding rate of 0.5 % are virtually indistinguishable from cover images when steganayzed using the feature-based method whereas for embedding rates higher than 5% of DCT coefficients, the feature-based steganalytic method works well in detecting MHPDM embedded stego images.

APPENDIX

MATLAB SOURCE CODE FOR MHPDM

The following computer program is the source code for the implementation of MHPDM (Histogram Preserving Data Mapping) in matlab.

```
% This function does a JPEG compression on a grayscale or RGB image
% Note that the Huffman compression is not JPEG standard (see PDF)
% Huffman code by Karl Skretting is used.
%
%%%%%%
% -Modified from JPEG code by Arno Swart, swart@math.uu.nl
%%%%%

function [msqe,compr]=jpg(file,scale)
% Add path to Huffman coder.
addpath('./huffman');
im = imread(file);
im = im2double(rgb2gray(im));
original=im;
figure(1);
imshow(im);
width = size(im,2);
height = size(im,1);
```

```
xblocks = width/8;

yblocks = height/8;

color=false;

numbits=width*height*8;

lumtable  =  lumquant(scale);

im = round(im*255);


% Y component uses luminance quantisation

dctcoef = jpgtrans(im,width,height,xblocks,yblocks,lumtable);

bitcount=huffcount(dctcoef,xblocks,yblocks);

disp(sprintf('Total bitcount %d',bitcount));

compr = 100-100*bitcount/numbits;

disp(sprintf('Compression: %f',compr));

msqe = sum(sum((original-im).^2))/(width*height);


%Find DCT of ‘f’

function F = dct(f)

        F=dct2(f);


%Find inverse DCT of ‘F’

function f = invdct(F)

        f=idct2(F);
```

```
function qtable  =  lumquant(scale)

          qtable=[   16,  11,  10,  16,  24,  40,  51,  61;

                    12,  12,  14,  19,  26,  58,  60,  55;

                    14,  13,  16,  24,  40,  57,  69,  56;

                    14,  17,  22,  29,  51,  87,  80,  62;

                    18,  22,  37,  56,  68, 109, 103,  77;

                    24,  35,  55,  64,  81, 104, 113,  92;

                    49,  64,  78,  87, 103, 121, 120, 101;

                    72,  92,  95,  98, 112, 100, 103,  99];

           qtable = round(qtable./scale);

           qtable(qtable<1) = 1;

           qtable(qtable>255) = 255;


%

% Standard JPG chrominance quantisation table

%

function qtable  =  chrquant(scale)

           qtable = [  17, 18, 24, 47, 99, 99, 99, 99;

                    18, 21, 26, 66, 99, 99, 99, 99;

                    24, 26, 56, 99, 99, 99, 99, 99;

                    47, 66, 99, 99, 99, 99, 99, 99;

                    99, 99, 99, 99, 99, 99, 99, 99;

                    99, 99, 99, 99, 99, 99, 99, 99;
```

```
              99, 99, 99, 99, 99, 99, 99, 99;
              99, 99, 99, 99, 99, 99, 99, 99];
        qtable = round(qtable./scale);
        qtable(qtable<1) = 1;
        qtable(qtable>255) = 255;
%
% Do zigzag ordering on matrix x, return vector x
%
function x = zigzag(x)
        zigzag= [  0,  1,  5,  6, 14, 15, 27, 28,
               2,  4,  7, 13, 16, 26, 29, 42,
               3,  8, 12, 17, 25, 30, 41, 43,
               9, 11, 18, 24, 31, 40, 44, 53,
              10, 19, 23, 32, 39, 45, 52, 54,
              20, 22, 33, 38, 46, 51, 55, 60,
              21, 34, 37, 47, 50, 56, 59, 61,
              35, 36, 48, 49, 57, 58, 62, 63]+1;

% convert matrices to column
        zigzag = zigzag(:);
        x = x(:);
 % do zigzag ordering
        x(zigzag) = x;
```

```
%

% Do 'inverse' zigzag ordering on vector x, return matrix x

%

function x = dezigzag(x)

            zigzag= [  0,  1,  5,  6, 14, 15, 27, 28,

                       2,  4,  7, 13, 16, 26, 29, 42,

                       3,  8, 12, 17, 25, 30, 41, 43,

                       9, 11, 18, 24, 31, 40, 44, 53,

                      10, 19, 23, 32, 39, 45, 52, 54,

                      20, 22, 33, 38, 46, 51, 55, 60,

                      21, 34, 37, 47, 50, 56, 59, 61,

                      35, 36, 48, 49, 57, 58, 62, 63]+1;

 % work on columns

            zigzag = zigzag(:);

            x = reshape(x(zigzag),8,8);

%

% Do a zero RLE on acoef, return arrays zerocounts and nonzeros.

%

function [zerocounts,nonzeros]=zerorle(acoef);

% Easiest is to first determine the nr of zeros at the end.

         k=31;

         while acoef(k)==0

               k=k-1;
```

```
    if k==0;
        zerocounts(1)=0;
        nonzeros(1)=0;
        break;
    end
end
curzerocount=0;
l=1;
for i=1:k
    if(acoef(i)==0)
        curzerocount=curzerocount+1;
        % Exception: 16 zeros
        if curzerocount==16;
            zerocounts(l)=15;
            nonzeros(l)=0;
            curzerocount=0;
            l=l+1;
        end
       else
        zerocounts(l)=curzerocount;
        nonzeros(l)=acoef(i);
        l=l+1;
        curzerocount=0;
```

```
          end
end
zerocounts(l)=0;
nonzeros(l)=0;


function dctcoef = jpgtrans(im,xsize,ysize,xblocks,yblocks,qtable)
        prevdc=0;
        chcoef=1;
          disp(sprintf('total number of 8*8 squares %d',xblocks*yblocks));
%JPG scan order is columns first
        for j=1:xblocks
             for i=1:yblocks
 % Extract 8x8 block
                imblock{i,j} = im((i-1)*8+1:(i*8),(j-1)*8+1:(j*8));
 % Shift 128 down
                imblock{i,j} = imblock{i,j} - 128;
 % Do a DCT
                dctcoef{i,j} = dct(imblock{i,j});


%organise coefficients with identical frequency into channels
 for ch=1:64
         switch ch
                     case {1,2,3,4,5,6,7}
```

```
  channel(ch,chcoef)=dctcoef{i,j}(1, mod(ch,8));

case 8

  channel(ch,chcoef)=dctcoef{i,j}(1,8);

case {9,10,11,12,13,14,15}

  channel(ch,chcoef)=dctcoef{i,j}(2, mod(ch,8));

case 16

   channel(ch,chcoef)=dctcoef{i,j}(2,8);

case {17,18,19,20,21,22,23}

  channel(ch,chcoef)=dctcoef{i,j}(3, mod(ch,8));

case 24

   channel(ch,chcoef)=dctcoef{i,j}(3,8);

case {25,26,27,28,29,30,31}

   channel(ch,chcoef)=dctcoef{i,j}(4, mod(ch,8) );

case 32

   channel(ch,chcoef)=dctcoef{i,j}(4,8);

case {33,34,35,36,37,38,39}

  channel(ch,chcoef)=dctcoef{i,j}(5, mod(ch,8) );

case 40

   channel(ch,chcoef)=dctcoef{i,j}(5,8);

case {41,42,43,44,45,46,47}

  channel(ch,chcoef)=dctcoef{i,j}(6, mod(ch,8) );

case 48

   channel(ch,chcoef)=dctcoef{i,j}(6,8);
```

```
                 case {49,50,51,52,53,54,55}
                    channel(ch,chcoef)=dctcoef{i,j}(7, mod(ch,8) );
                 case 56
                     channel(ch,chcoef)=dctcoef{i,j}(7,8);
                 case {57,58,59,60,61,62,63}
                    channel(ch,chcoef)=dctcoef{i,j}(8, mod(ch,8) );
                 case 64
                     channel(ch,chcoef)=dctcoef{i,j}(8,8);
                otherwise
                    disp('wrong ch');
            end
        end
         chcoef=chcoef+1;
     end
 end
for j=1:xblocks
     for i=1:yblocks
         % Quantize
         dctcoef{i,j} = round(dctcoef{i,j}./qtable);
         % Differential code the DC
         temp = dctcoef{i,j}(1,1);
         dctcoef{i,j}(1,1) = dctcoef{i,j}(1,1) - prevdc;
         prevdc=temp;
```

```
      % Zigzag
      dctcoef{i,j} = zigzag(dctcoef{i,j});
    end
 end
%
% Inverse JPEG procedure
%
function img = jpginvtrans(dctcoef,xsize,ysize,xblocks,yblocks,qtable)
      prevdc=0;
      for j=1:yblocks
    for i=1:xblocks
      % De-zigzag
      dctcoef{i,j} = dezigzag(dctcoef{i,j});
      %Un-differential code the DC
      dctcoef{i,j}(1,1) = dctcoef{i,j}(1,1) + prevdc;
      prevdc=dctcoef{i,j}(1,1);
     % De-Quantize
      dctcoef{i,j} = dctcoef{i,j}.*qtable;
      % Do an inverse DCT
      imblock{i,j} = invdct(dctcoef{i,j});
      % Shift 128 up
      imblock{i,j} = imblock{i,j} + 128;
      %Rebuild the image

      img((i-1)*8+1:i*8,(j-1)*8+1:j*8) = imblock{i,j};

    end

end


%

% Count the nr. bits that a Huffman coding would take.

%

function bitcount=huffcount(dctcoef,xblocks,yblocks)

      l=1;

      bitcount=0;

      for j=1:yblocks

            for i=1:xblocks

      % The dccoef's are treated separately.

                  dccoef(l)=dctcoef{i,j}(1);

                  l=l+1;

      % Get the ac coef's

                  accoef = dctcoef{i,j}(2:32);

                   [zerocounts,nonzeros]=zerorle(accoef);

                  bitcount=bitcount+sum(hufflen(zerocounts));

                  bitcount=bitcount+sum(hufflen(nonzeros));

            end
    end
    bitcount=bitcount+sum(hufflen(dccoef));


%
%Hide binary message in channels 1 to 21 of the image
%Using the HPDM algorithm, receive the relative
%entropy values between the original and embedded
%channels and plot them  against the channel numbers.
%
function hpdm0422_plot (channel)
    [ro(1),re(1),rs(1)]=mapping0422(channel(2,:));
    [ro(2),re(2),rs(2)]=mapping0422(channel(9,:));
    [ro(3),re(3),rs(3)]=mapping0422(channel(17,:));
    [ro(4),re(4),rs(4)]=mapping0422(channel(10,:));
    [ro(5),re(5),rs(5)]=mapping0422(channel(3,:));
    [ro(6),re(6),rs(6)]=mapping0422(channel(4,:));
    [ro(7),re(7),rs(7)]=mapping0422(channel(11,:));
    [ro(8),re(8),rs(8)]=mapping0422(channel(18,:));
    [ro(9),re(9),rs(9)]=mapping0422(channel(25,:));
    [ro(10),re(10),rs(10)]=mapping0422(channel(33,:));
    [ro(11),re(11),rs(11)]=mapping0422(channel(26,:));
    [ro(12),re(12),rs(12)]=mapping0422(channel(19,:));
    [ro(13),re(13),rs(13)]=mapping0422(channel(12,:));
```

```
[ro(14),re(14),rs(14)]=mapping0422(channel(5,:));

[ro(15),re(15),rs(15)]=mapping0422(channel(6,:));

[ro(16),re(16),rs(16)]=mapping0422(channel(13,:));

[ro(17),re(17),rs(17)]=mapping0422(channel(20,:));

[ro(18),re(18),rs(18)]=mapping0422(channel(27,:));

[ro(19),re(19),rs(19)]=mapping0422(channel(34,:));

[ro(20),re(20),rs(20)]=mapping0422(channel(41,:));

[ro(21),re(21),rs(21)]=mapping0422(channel(49,:));

figure(1);

title('hpdm');

i=1:21;

%plot (i,r(i),'bo-');

%plot(i,ro,'b-',i,re,'g-',i,rs,'r-');

plot(i,rs,'r-');

title('Relative entropy between original and embedded subchannel using
HPDM (results for "lenna.tiff")');

xlabel('subchannel number');

ylabel('relative entropy');

axis([0,25,0,0.12]);
```

```
        %
        % This function implements Egger's Histogram Preserving Data Mapping
        % algorithm(HPDM) as described in [1]
        %
        function [resultodd,resulteven,result]=mapping0422(x)
        %calculates the dynamic range of the input data
        dyran=round(max(x)-min(x)+1);
        %histogram of original subchannel
        [num1,pos1]=hist(x,dyran);
        length(pos1);


         if(rem(length(x),2)==0)
                i=1:length(x)/2;
                j=length(x)/2+1:length(x);
        else
                i=1:length(x)/2-0.5;
                j=length(x/2)+0.5 : length(x);
        end
 %s is the binary message to be embedded into the original subchannel.IT %contains
equal number of zeros and ones.
        s(i)=0;
        s(j)=1;
```

```
is1=find(s==1);
is0=find(s==0);
if rem(dyran,2)==0
        numodd=1:2:dyran-1;
         numeven=2:2:dyran;
 else
        numodd=1:2:dyran;
        numeven=2:2:dyran-1;
 end
 % x2 is the DCT coefficients in which a '1' is to be embedded
x2=x(is1);
[num12,pos12]= hist(x2,round(max(x2)-min(x2)+1));
i=1;
for j=1:1:length(num12)
         for a=num12(j)-1:-1:0
                 xn12(i)=pos12(j);
                 i=i+1;
         end
end
 an2 = rand(1,sum(num12));
  j=1:length(x2);
 xn22(j)= xn12(j)-an2(j);
 n22=[0,ones(1,length(x2))];
 n22=cumsum(n22)./sum(n22);
 xn22=sort(xn22);
 xn22=[xn22(1)-1,xn22];
% split the histogram into odd and even bin positions
num1even=num1(numeven);
num1even(1)=1;
pos1even=pos1(numeven);
stepyeven=(cumsum(num1even)./sum(num1even));
%Canculate the thresholds, Teven for mapping onto the even histogram by
%interpolation
Teven=interp1q(n22(:),xn22(:),stepyeven(:));
% messase embedding
for i=1:length(xn22)
        if xn22(i)<= Teven(1)
                y2(i)=pos1even(1);
        else
        if xn22(i)>Teven(end)
                   y2(i)= pos1even(end);
        else
                for j=2:length(Teven)
                        if xn22(i) > Teven(j-1) && xn22(i) <= Teven(j)
                                y2(i)= pos1even(j);
                        end
                end
            end
    end
end
%even histogram after embedding
[numy2,posy2] = hist(y2,pos1even);
% x1 is the DCT coefficients in which a '0' is to be embedded
x1=x(is0);
[num11,pos11]= hist(x1,round(max(x1)-min(x1)+1));
i=1;
for j=1:1:length(num11)
        for a=num11(j)-1:-1:0
                xn11(i)=pos11(j);
                i=i+1;
        end
end
 an1 = rand(1,sum(num11));
 j=1:length(x1);
 xn21(j)= xn11(j)-an1(j);
 n21=[0,ones(1,length(x1))];
 n21=cumsum(n21)./sum(n21);
 xn21=sort(xn21);
 xn21=[xn21(1)-1,xn21];
```

```
num1odd=num1(numodd);

pos1odd=pos1(numodd);

stepyodd=(cumsum(num1odd)./sum(num1odd));

%Canculate the thresholds, Todd for mapping onto the odd histogram by

%interpolation

Todd=interp1q(n21(:),xn21(:),stepyodd(:));

% message embedding

for i=1:length(xn21)

    if xn21(i)<= Todd(1)

        y1(i)=pos1odd(1);

    else

    if xn21(i)>Todd(end)

        y1(i)= pos1odd(end);

    else

        for j=2:length(Todd)

            if xn21(i) > Todd(j-1) && xn21(i) <= Todd(j)

                y1(i)= pos1odd(j);

            end

          end

    end

  end

end
```

```
%odd histogram after embedding
[numy1,posy1] = hist(y1,pos1odd);
% The combined odd and even coefficients
y=[y2,y1];
length(y);
%the combined odd and even histogram after embedding the message
[numy,posy] = hist(y,pos1);
%calculates pdf of original and embedded suchannels
pdfx=num1/sum(num1);
pdfy=numy/sum(numy);
my_plot_y=[pdfx;pdfy]';
my_plot_h=stem(pos1,my_plot_y,':');
set(my_plot_h(1),'MarkerFaceColor','blue')
set(my_plot_h(2),'MarkerFaceColor','red','Marker','square')
%relative entropy calculation between the original histogram and the
%histogram after emedding
   P1=num1odd/sum(num1odd);
   Q1=numy1/length(y1);
  Px1 = P1 + eps;
  Qx1 = Q1 + eps;
  %--------------------
  resultodd=sum( Px1.*( (log2(Px1./Qx1)) ) )
  %-------------------------------------
  P2=num1even/sum(num1even);
   Q2=numy2/length(y2);
  Px2 = P2 + eps;
  Qx2 = Q2 + eps;
  %---------------------
  resulteven=sum(  Px2.*( (log2(Px2./Qx2)) )  )
  %----------------------------------------
Px3=num1/sum(num1);
 Qx3=numy/sum(numy);
 Px3 = Px3 + eps;
  Qx3 = Qx3 + eps;
  %---------------------
  result=sum(  Px3.*( (log2(Px3./Qx3)) )  )
  %----------------------------------------
```

REFERENCES

1. Eggers, J.J., Bauml, R., & Girod, B. (2002). A communications approach to image steganography. Proc. of SPIE vol. 4675, Security and Watermarking of Multimedia Contents IV.

2. Cachin, C. (1998). An information-theoretic model for steganography. Proc. of 2nd Workshop on Information Hiding, vol. 1525, Lecture Notes in Computer Science.

3. Chen, B., & Wornell, G.W. (1998). Digital watermarking and information embedding using dither modulation. Proc. of IEEE Workshop on Multimedia Signal Processing (MMSP-98), 273-278.

4. Farid, H. (2002). Detecting hidden messages using higher-order statistical models.ICIP2002.

5. Lyu, S., & Farid, H. (2002). Detecting hidden messages using higher-order statistics and support vector machines. Proc. of 5th International Workshop on Information Hiding.

6. Martin, A., Sapiro, G., & Seroussi, G. (2004). Is image steganography natural? Information Theory Research Group, HP Laboratories Palo Alto, HPL-2004-39.

7. Tzschoppe, R., Bauml, R., Huber, J.B., & Kaup, A. (2003). Steganographic system based on higher-order statistics. Proc. of SPIE vol. 5020, Security and Watermarking of Multimedia Contents V.

8. Chen, B., & Wornell, G.W. (2001). Quantization Index Modulation: A class of provably good methods for digital watermarking and information embedding. IEEE Transactions on Information Theory, vol. 47, no. 4.

9. Fridrich, J. (2004). Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes. Proc. of 6th International Workshop on Information Hiding, 67-81.

10. Westfeld, A., & Pfitzmann, A. (2000). Attacks on steganographic systems. 3rd International Workshop. Lecture Notes in Computer Science, vol. 1768, 61-75.

11. Fridrich, J., Goljan, M., Hongea, D., & Soukal, D. (2003). Quantitative steganalysis: Estimating secret message length. ACM Multimedia Systems Journal. Special Issue of Multimedia Security, vol. 9, 288-302.

12. Farid, H., & Siwei, L. (2002). Detecting hidden messages using higher-order statistics and support vector machines. Information Hiding, 5th International Workshop. Lecture Notes in Computer Science, vol. 2578, 340-354.

13. Avcibas, I., Memon, N., & Sankur, B. (2001). Steganalysis using quality metrics. SPIE Security and Watermarking of Multimedia Contents II, electronic Imaging.

14. Provos, N. (2001). Defending against statistical steganalysis. 10th USENIX Security Symposium.

15. Sallee, P. (2003).Model based steganography. International Workshop on Digital Watermarking, 174-188.

16. Westfeld, A., & Pfitzmann, A. (2001). High capacity despite better steganalysis (F5-A steganographic algorithm). Information Hiding. 4th International Workshop. Lecture Notes in Computer Science, vol. 2137, 289-302.

17. Sallee, P. (2004). Model-based methods for steganography and steganalysis. International Journal of Image and Graphics. Special Issue on Image Data Hiding.

18. Pfitzmann, B. (1996). Information hiding terminology. Information Hiding, First International Workshop, vol. 1174, 374-350

19. Kharrazi, M. (2005). Benchmarking steganographic and steganalysis techniques. SPIE Security, Steganography, and Watermarking of Multimedia Contents VII, 252-263